# BEEBLET

$3-00

ISSN 0112-3351

VOLUME 3 NUMBER 1 FEBRUARY 1985

**BBC/ACORN**

COMPUTER USER GROUP
NZ (inc)

# CONTENTS

Editorial 3
REcursion Answer 4
Improved 50 Baud Teleprinter Driver 8
Windows 16
Crossword Solution 19
Membership Map 20
Help Wanted 22
Hints 23
Lost Memory? 24
Editing BASIC in View 25
Sort Report 26
Review: Merlin Scribe 29
Cartoon 33
Review: Aviator 35
VDU5 for Beginners to Graphics 37
Classified Advertisements 38
Hall of Fame 39
Local Groups 40

ADVERTISER'S INDEX:

APTEC 28
Einstein Scientific 34

# EDITORIAL

Bet you've been wondering what on earth has happened to your copy of Beeblet? Supposed to be earlier this month wasn't it? Welllllll..

First of all, Wellington has just had its most brilliant summer in some years. The Beeb came off second best. Secondly, the Editor got clobbered by the bug currently doing the rounds, over the weekend assigned for Beeblet production. Thirdly, there were a number of distractions, who will remain nameless (but never-the-less delightful). By now some of you will wondering how come how come Beeblet emerged at all. Well, when your Editor staggered in off the late ferry from Picton ( a wonderful and much needed, but very inexpensive, weekend at the Gem resort ), a little note from the previous Editor was sitting on the dining room table...

The relevent extract reads as follows:
"..
2) "Reward" for finishing magazine? Supper? Breakfast? Meat pie and apple crumble in fridge. Medium oven 20 mins......" Well, I had to finish it then - didn't I! Thanks very much Anne - much appreciated.

**********************

Apologies to those whose contributions didn't make this issue. I'm now working on the March issue...

**********************

You will notice the "measles" chart of membership distribution which David Fung kindly agreed to prepare. This was correct at 31 December 1984. Not shown are 1 school membership in Auckland (special school rate here?), 1 Australian based and 1 U.K. based member.

**********************

### A.G.M.

Members should soon receive notification of the next A.G.M. together with details of a small subsidy (based on membership numbers) available to local groups towards sending a representive to the AGM. Remember, this is not supposed to be just a Wellington affair - now is the time to work out those issues of importance to you, and check the constitution re necessary circulation of notices of motions etc.

# RECURSION ANSWER

G.C.F

Somebody in the December issue wanted to know how to represent a recursive function in a structured diagram. This article provides a method, and an example program and diagram, but first a definition, and a Government Health Warning!

## DEFINITION:

The best definition of recursion I've come across is in the glossary of some book (title escapes me), and goes like this:
**recursion** - see recursion
It's not quite like an endless loop. Recursion occurs if a procedure calls itself. Each time the procedure is called, it must find extra memory to save its local variables and store its return address. Formal parameters (the variables in brackets aftre the PROC name) also count as LOCAL variables. Recursion can thus rapidly eat in up huge quantities of memory whilst the code remains looking innocently compact.

Recursion can be quite complex, with a procedure calling other procedures which in turn call the first procedure. Functions can also be recursive, and there is no reason why a recursive procedure cannot call a recursive function...

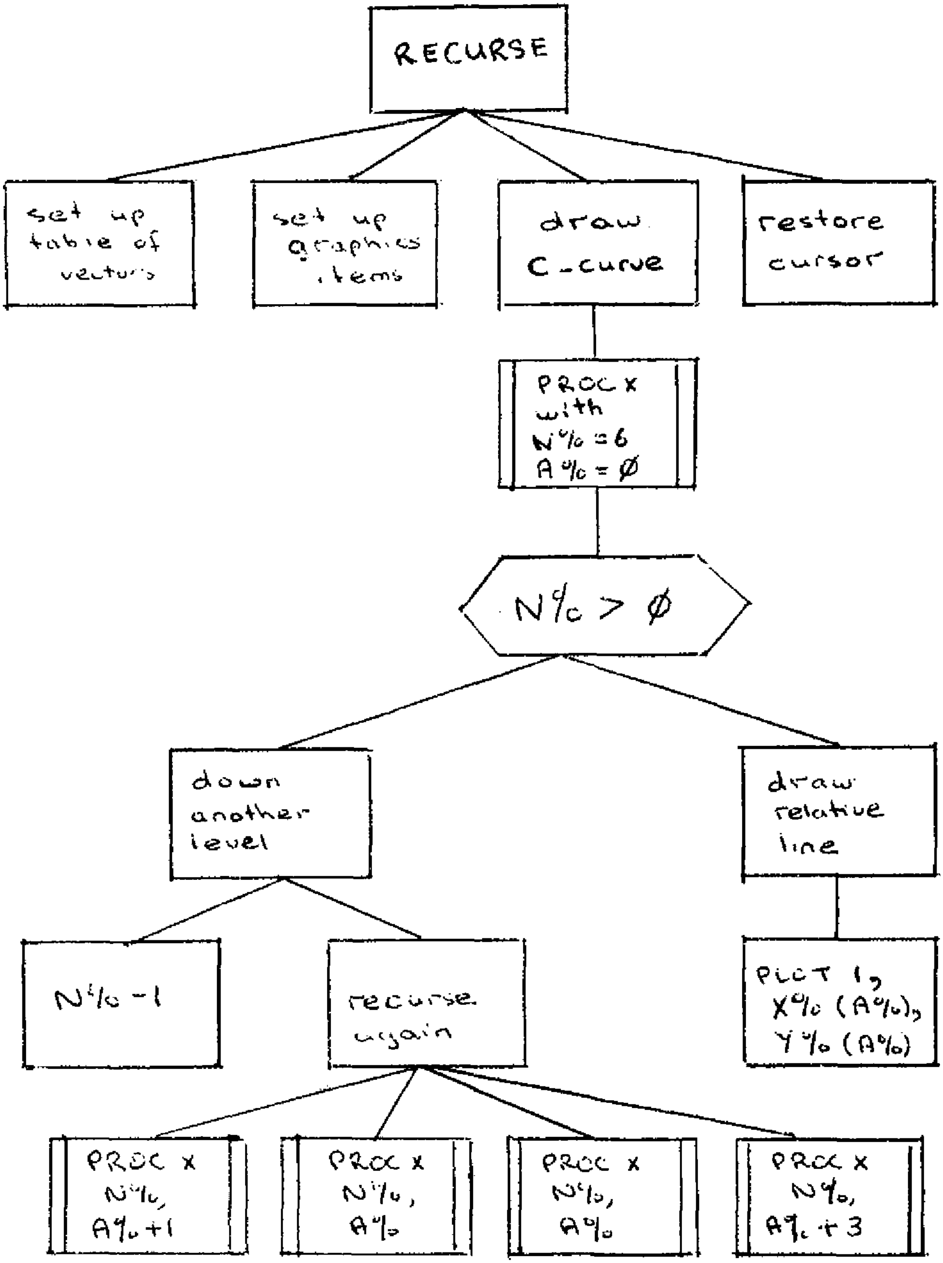
WARNING:
Excessive recursion may not lead to blindness (not proven), but it can soak up vast amounts of your time trying to derive the appropriate logic and immense amounts of time trying to debug the resultant code.

The Golden Rule of recursion is:
"Never use recursion if there is a simpler iterative solution." The practical meaning of this is that you should be looking for solutions in terms of loops, using FOR..NEXTs, and Repeat-Untils first, before considering recursion.

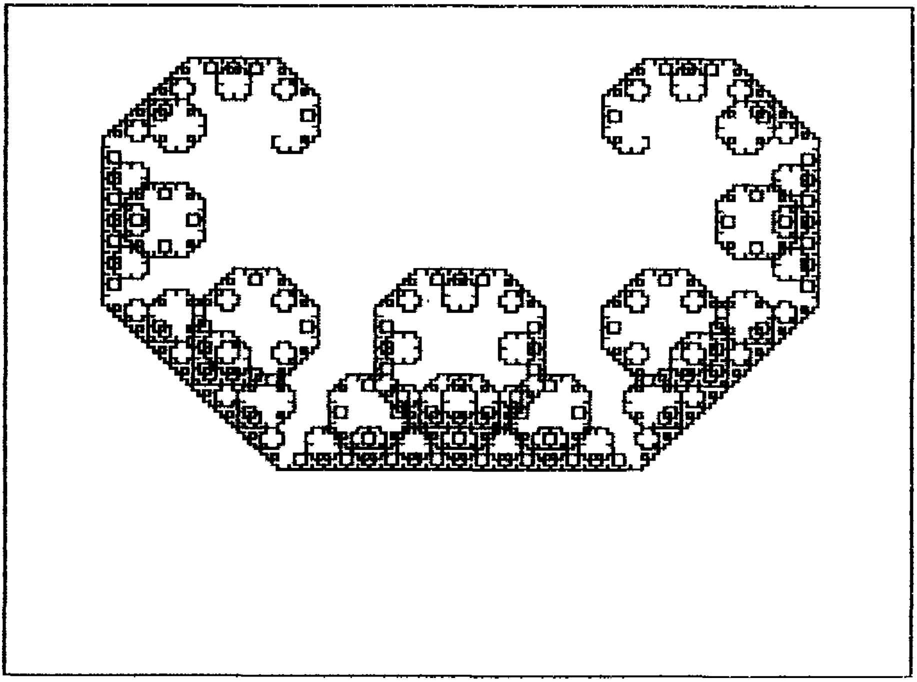
If you persist in wanting to use recursion, be sure that the PROC will stop recursing at some point, before you exhaust available memory and blow the PROC/FN stack. This is one of the reasons you must never GOTO out of a PROC - the return address is left on the PROC stack, which is normally used up be the ENDPROC - the same applies to Functions (which end in "=..." not ENDPROC).

## STRUCTURE DIAGRAM

This is a complete, if somewhat terse, diagram of the whole program. The boxes with double lines indicate PROC calls. From the breakdown of PROCX (level 3) it can be seen that it calls itself as part of its


RECURSE
set up table of vectors
set up graphics items
draw C-curve
restore cursor
PROC X with N% = 6 A% = 0
N% > 0
down another level
draw relative line
N% -1
recurse again
PLOT 1, X%(A%), Y%(A%)
PROC X N%, A%+1
PROC X N%, A%
PROC X N%, A%
PROC X N%, A%+3

definition (level 7).



PROGRAM COMMENTS:
The program draws a picture which looks like an ornate letter "C" lying on its side. It is based on the listing on p.79 of "Creative Graphics" by John Cownie (pub. Acornsoft). The book has many useful graphics techniques, though some of the programming style is sometimes a little clumsy. The version here is much faster though!

First the program sets up and fills the X,Y components of the relative vectors it is going to use later - COS and SIN functions are slow, so it saves time to do the calculations once and refer to them in a table later.

The VDU19 redefines logical colour 1 to yellow, and VDU5 removes the flashing cursor. To draw relative to the previous point PLOT 1 is used.

The real work is done in PROCX, which calls itself (this is called recursion) several times. The depth of recursion is given by N%.

If the MOD operator had been implemented correctly (it should always return a positive value), it would not be necessary to add 3 to subtract 1. This is because normally (N+3)MOD4 = (N-1)MOD4
eg (2+3)MOD4 = (5)MOD4 = (4+1)MOD$ = 1 = (2-1)MOD4

```
   10 REM  RECURSE
   20 REM  A graphic example of recursion, "C - curve".
   30 DIM X%(3)
   40 DIM Y%(3)
   50 P = PI/2
   60 FOR A%= 0 TO 3
   70   X%(A%) = 8*COS(A%*P)
   80   Y%(A%) = 8*SIN(A%*P)
   90   NEXT A%
  100 MODE 4
  110 VDU 19,1,3;0;
  120 VDU 5
  130 MOVE 380,200
  140 PROCX(6,0)
  150 VDU 4
  160 END
  170
  180 DEF PROCX(N%,A%)
  190 IF N% > 0 THEN N% = N%-1 :PROCX(N%,A%+1) :PROCX(N%,A%) :PROCX(N%,A%) :PROC
X(N%,A%+3) ELSE A%=A%MOD4 :PLOT 1,X%(A%),Y%(A%)
  200 ENDPROC
```

## RECURSION CHALLENGE?

A) Can you find a simple non-recursive definition of the following function?
B) Can you find a formula that gives the maximum value for a given M%, provided that N% is positive and not greater than M%?
C) Can you predict the level of recursion for a given M% and N%?

```
   10 REM  RECURFN - Recursive function
   20 FOR I% = 1 TO 6
   30   FOR J% = 1 TO I%
   40     PRINT"FNa(";I%;",";J%;") = ";FNa(I%,J%)
   50     NEXT J%
   60   NEXT I%
   70 END
   80
   90 DEF FNa(M%,N%)
  100 IF M% > 1 AND N% > 1 THEN = FNa(M%DIVN%,N%) + FNa(M%,M%MODN%) ELSE = 1
```

# IMPROVED 50 BAUD TELEPRINTER DRIVER

by Warren W.

In the May and June 1983 issues of this magazine we ran a project involving both the software and hardware needed to drive a 50 Baud Telex machine from the BBC Microcomputer. The software, as presented, had a few shortcomings which proved annoying in practice:

1. The CPU was tied up full time doing the timing for the serial output to the printer.
2. There was no buffering at all, and hence even very small print jobs tied up the computer immediately.
3. The driver required user defined function keys to enable and disable it, instead of the usual CTRL B and CTRL C.
4. Printer control from within a program required the use of special PROCedures, rather than the normal VDU 2 and VDU 3 commands. It would thus not work with standard word processors and the like.
5. The driver would only work with operating system version 0.1.

The listing that follows solves all of these problems. Firstly, it uses one of the timers in the user/printer 6522 versatile interface adapter (VIA) to control the timing of the serial output to the printer. Secondly, the printer driver is implemented as a user supplied printer driver (see BBC User Guide, page 408), which means that all characters for printing are sent to the normal printer buffer. Also the normal printer controls can be used following the execution of *FX5,3. Finally, the program is operating system independent in that the program which generates the printer driver actually reads the required vector value from RAM.

The listing is broken up into several parts which we will look at briefly:

Lines 10 to 80 give title information.

Lines 120 to 220 define some of the function keys. Since I always load and run this program when I first turn the computer on, I find it useful to include general purpose key definitions. These have nothing whatsoever to do with the printer driver. You can include your own favourites for f8 and f9.

Lines 260 to 280 read the IRQ2V vector address.

Lines 290 to 330 label and initialise zero page addresses that are used by the routines as counters or flags.

Lines 350 to 390 label addresses of registers within the VIA which controls the user and printer ports.

Lines 440 to 710 assemble a routine which is located on zero page at address &0075. This routine is entered on interrupt. If the interrupt has originated from the timer, and there are more bits to send, then the next bit is clocked out via the printer port and the timer is reset for another 20ms countdown. At the end of the 20ms period an interrupt is generated and the routine is entered again. The area of memory used to store this routine (&0075 to &00A7) is partly free, partly used by Econet and partly by the non-maskable interrupt routine (NMI). So if you are using Econet or discs or paged ROMs then the routine is likely to be corrupted, and you will therefore need to locate the routine at &09C0, which is normally used as a cassette output buffer or a speech buffer. You will need to change lines 450, 1990 and 2000.

Lines 750 to 1890 assemble the printer driver routine which is located at address &0D00. This routine is entered automatically every 10ms so long as there is at least one character in the printer buffer, i.e. the printer driver is active. When the printer buffer is emptied the driver is sent dormant and is not entered again until a further character is inserted in the buffer.

On entry the driver checks that the call is intended for it and makes sure that the last character is not still being sent. If the last character sent was a shift control then the next character is retrieved from temporary storage, otherwise the next character is extracted from the printer buffer. Special checks and subroutines deal with shifts, carriage returns, and the like, and the character is then converted from ASCII to Telegraph Code via a lookup table. Start and stop bits are added and the character is loaded into the printer port. The timer controlling the outpulsing is initialised and' control is finally returned to the operating system. This routine is stored on page &0D00 which is normally free on machines which do not use paged ROMs. On disc based systems you could use page &0A00 which is normally used as a cassette input buffer or RS 423 port buffer. You will need to change lines 760, 1930 and 1940.

Lines 1930 to 2000 set the new vectors for accessing the printer driver and interrupt routines, and set up the registers of the user/printer VIA to the requierd status.

Lines 2035 to 2070 select the user printer driver and screen for output, set the suppressed print character, and cause the IRQ1V interrupt routine to pass user/printer VIA interrupts on to the printer driver.

Lines 2110 to 2130 send a test message to the printer.

Lines 2180 to 2300 define the FuNction that generates the character conversion lookup table. This function is called from line 1870.

If you would like a cassette copy of this program please send a cassette, together with return postage, to: Printer Driver Program, Warren W, P.O. Box 9592, Wellington.

## DO YOU WANT TO ...

* use a teleprinter with your Beeb...

* acquire PROCs for your library...

* have some NON-alien games...

* understand ENV., make joysticks...

**1983** issues

**BARGAIN** **50** c each

July and December already sold out, so hurry...

take pot luck, send up to $5 and get up to 10 issues

(eg $3 for 6 issues - all different!)

The Secretary, BBC/ACORN Computer User Group NZ Inc,
P.O. Box 9592, Wellington.

```
 10 REM ****************************
 20 REM *  USER SUPPLIED PRINTER   *
 30 REM *    DRIVER FOR SENDING    *
 40 REM *  50 BAUD TELEGRAPH CODE  *
 50 REM *    TO A TELEPRINTER      *
 60 REM *      by W P WILLIS       *
 70 REM * Ver 1.0  OS 1.2  1/11/84 *
 80 REM ****************************
 90 REM
100 REM ** KEY DEFINITIONS *********
110 REM
120 *KEY 0 |OCLS|MRUN|M
130 *KEY 1 MO.7|ML.08|M|NL.|M
140 *KEY 2 W.69:MO.7|ML.07|M|OL.|M
150 *KEY 3 *C.|M
160 *KEY 4 SA."
170 *KEY 5 CH.""|M
180 *KEY 6 *LO.""8000|M
190 *KEY 7 MO.7|M
220 *KEY 10 OLD|MRUN|M
230 REM
240 REM ** LABEL AND INITIALISE ADDRESSES **
250 REM
260 lobyte=?&0206:REM LSB of IRQ2V vector
270 hibyte=?&0207:REM MSB of IRQ2V vector
280 irq2v=(hibyte*256)+lobyte:REM 2-byte IRQ vector
290 copy=&0070:?&0070=0:REM Stores copy of code to be sent
300 ltrsfigsflag=&0071:?&0071=0:REM Flag for current shift mode
310 bitcount=&0072:?&0072=0:REM No. of bits yet to be sent
320 shiftflag=&0073:?&0073=0:REM Flag if last character was a shift
330 crflag=&0074:?&0074=0:REM Flag if last character was a CR
340 REM
350 port=&FE61:?&FE61=&FF:REM Address of printer port
360 timerlo=&FE64:REM LSByte of Timer1 latch
370 timerhi=&FE65:REM MSByte of Timer1 latch
380 status=&FE6D:REM User VIA interrupt status register
390 osbyte=&FFF4:REM OSBYTE routine address
400 REM
410 REM ** ASSEMBLE PRINTER INTERRUPT ROUTINE **
420 REM ** FOR SENDING BITS AT 2MS INTERVALS ***
430 REM
440 FOR pass=0 TO 3 STEP 3
450 P%=&0075
460  [
470 OPT pass
480 .printerrupt
490 PHA:LDA &FC    \Preserve registers
500 PHA:TXA:PHA
```

```
510 TYA:PHA
520 LDA status    \Get user VIA interrupt status
530 AND #&C0      \Mask bits of no interest
540 CMP #&C0      \Is it a Timer1 interrupt?
550 BNE end       \No - exit
555 STA status    \Clear user VIA interrupt status register
560 LDA bitcount  \Test bit counter for more bits to send
570 BEQ end       \No = exit
580 DEC bitcount  \Send next bit
590 SEC           \(Send 1's for CR delay)
600 ROR port
610 LDA #&CD      \Load time interval low byte
620 STA timerlo
630 LDA #&4D      \Load time interval high byte
640 STA timerhi
650 .end
660 PLA:TAY       \Restore registers
670 PLA:TAX:PLA
680 STA &FC:PLA
690 JMP irq2v     \Hand back to interrupt routine
700 ]
710 NEXT pass
720 REM
730 REM ** ASSEMBLE USER PRINTER DRIVER **
740 REM
750 FOR pass=0 TO 3 STEP 3
760 P%=&0D00
770 [
780 OPT pass
790 .start
800 PHA:TXA:PHA    \Preserve contents of A and X
810 TYA:PHA:PHP    \Preserve contents of Y and P
820 CPX #&03       \Test for printer buffer
830 BNE end2
840 CPY #&03       \Test for user printer
850 BNE end2
860 LDA bitcount   \Test for character still being sent
870 BNE end2
880 LDA shiftflag  \Test for previous character = shift
890 BNE noshift
900 .getbuffer
910 LDX #&03       \Get a character from the printer buffer
920 LDA #&91
930 SEI            \Set interrupt disable
940 JSR osbyte
950 CLI            \Clear interrupt disable
960 BCS dormant    \Go dormant if printer buffer empty
970 TYA            \Put character from buffer into A
```

```
 980 JSR astotg      \Translate from ASCII to Telegraph Code
 990 STA copy        \Store a copy of the character
1000 .shiftest
1010 AND #&20        \Test bit 5 to see if figures or letters
1020 CMP ltrsfigsflag \Test to see if shift needed
1030 BEQ getstore    \Get character from store if no shift required
1040 TAX             \Preserve character
1050 LDA #&01        \Set shift flag
1060 STA shiftflag
1070 TXA             \Restore character
1080 CMP #&20        \Test for figures shift required
1090 BEQ figures
1100 .letters
1110 LDA #&00        \Clear letters/figures flag
1120 STA ltrsfigsflag
1130 LDA #&1F        \Send letters shift
1140 JMP sendchar
1150 .settimer2
1160 DEC bitcount
1170 LDA #&CD        \Load low byte of Timer1 latch
1180 STA timerlo
1190 LDA #&4D        \Load high byte of Timer1 latch
1200 STA timerhi
1210 .end2
1220 CLC             \Clear carry to keep printer driver active
1230 .end3
1240 PLP:PLA:TAY     \Restore P and Y registers
1250 PLA:TAX:PLA     \Restore X and A registers
1260 RTS             \Return control to Operating System
1270 .dormant
1280 LDX #&03        \Send printer driver dormant
1290 LDA #&7B
1300 JSR osbyte
1310 SEC             \Set carry to keep driver dormant
1320 JMP end3
1330 .figures
1340 LDA #&20        \Set letters/figures flag
1350 STA ltrsfigsflag
1360 LDA #&1B        \Send figures shift
1370 JMP sendchar
1380 .noshift
1390 LDA #&00        \Clear shift flag
1400 STA shiftflag
1410 .getstore
1420 LDA copy        \Get stored character
1430 .sendchar
1440 ORA #&E0        \Add stop bits
1450 ASL A           \Add start bits
```

```
1460 STA port       \Load character into printer port
1470 LDA #&08       \Test for carriage return
1480 CMP crflag
1490 BEQ longdelay
1500 STA bitcount   \Set bit counter
1510 JMP settimer2
1520 .longdelay
1530 LDA #&00       \Clear carriage return flag
1540 STA crflag
1550 LDA #&20       \Set large bitcount for long delay
1560 STA bitcount
1570 JMP settimer2
1580 .astotg        \Convert ASCII to Telegraph Code
1590 AND #&7F       \Mask off most significant bit
1600 CMP #&0D       \Test for carriage return
1610 BEQ carreturn
1620 CMP #&0A       \Test for line feed
1630 BEQ linefeed
1640 CMP #&7F       \Test for backspace and delete
1650 BEQ backspace
1660 CMP #&20       \Test for control codes
1670 BMI backspace
1680 CMP #&60       \Test for lower case
1690 BMI mask
1700 SEC            \Convert lower to upper case
1710 SBC #&20
1720 .mask
1730 AND #&3F       \Mask off 2 most significant bits
1740 TAX
1750 LDA attbl,X    \Get Telegraph Code from lookup table
1760 RTS
1770 .carreturn
1780 LDA #&08       \Load code for carriage return
1790 STA crflag     \Set carriage return flag
1800 RTS
1810 .linefeed
1820 LDA #&02       \Load code for line feed
1830 RTS
1840 .backspace
1850 LDA #&00       \Load code for space
1860 RTS
1870 OPT FNtable    \Generate lookup table for code conversion
1880 ]
1890 NEXT pass
1900 REM
1910 REM ** SET UP USER VIA REGISTERS AND MODIFY VECTORS
1920 REM
1930 ?&0222=&00:REM Establish user printer vector
```

```
1940 ?&0223=&0D
1950 ?&FE63=&FF:REM Set printer port for output
1960 ?&FE6B=&00:REM Set auxiliary control register in VIA
1970 ?&FE6E=&3F:REM Clear bits 1-6 in control register
1980 ?&FE6E=&C0:REM Set bits 7-8 in control register
1990 ?&0206=&75:REM Set interrupt vector for printer interrupt routine
2000 ?&0207=&00
2010 REM
2020 REM ** SELECT USER PRINTER AND ENABLE USER VIA INTERRUPTS **
2030 REM
2035 REM Select user printer driver
2040 *FX 5,3
2045 REM Select printer and screen for output
2050 *FX 3,0
2055 REM Set suppressed character to be &00
2060 *FX 6,0
2065 REM Mask user 6522 interrupts from OS
2070 *FX 231,0
2080 REM
2090 REM ** PRINT MESSAGE **
2100 REM
2110 VDU 2     :REM Turn printer on
2120 PRINT"Hi there! This is your printer printing."
2130 VDU3      :REM Turn printer off
2140 END
2150 REM
2160 REM ** CREATE LOOKUP TABLE FOR CODE CONVERSION **
2170 REM
2180 DEF FNtable
2190 attbl=P%
2200 FOR I%=1 TO 64
2210 READ code$
2220 ?P%=EVAL("&"+code$)
2230 P%=P%+1
2240 NEXT I%
2250 RESTORE
2260 DATA 3A,03,19,0E,09,01,0D,1A,14,06,0B,0F,12,1C,0C,18,16,17,0A
2270 DATA 05,10,07,1E,13,1D,15,11,2F,04,32,04,04,04,3C,25,31,34,2D
2280 DATA 04,25,2F,32,31,31,2C,23,3C,3D,36,37,33,21,2A,30,35,27,26
2290 DATA 38,2E,2E,04,3E,04,39
2300 =pass
```

# WINDOWS

Anne F.

Windows are the thing to sell micros these days; it is the latest 'gimmick', so who would have thought that the comparatively out-dated Beeb would have this sophistication also? It does its best, though unlike the latest machines, can only have one text window and one graphics window active at a time. If you wonder why you've never noticed any windows on your Beeb, it's because unless programmed otherwise, the two windows overlap, and take up the whole screen. Commercial programs often come in two parts. The first sets up on the screen a company logo and a small text window. It is in this window that the loading messages appear while the second part of the program loads in. The obvious function of a window is to keep its contents separate from whatever else is happening on the screen, and prevent them from interfering with the rest of the action.

## TEXT WINDOWS:

A text window is defined within the limits of the MODE you will be using, and in terms of characters across and lines down. These equate directly with the TAB statement. The default window in Mode 0 is the equivalent of defining a window with TAB(0,31) as bottom left corner and TAB(79,0) as top right corner.

A window is defined via VDU28, so to define a Mode 0 text window incorporating the whole screen would require the following command:
VDU28,0,31,79,0
So when you redefine a text window you are resetting the screen size that can be used for text. The PRINTTAB command is restricted by the size of the window. TAB(0,0) is no longer at the very top of the screen, but could be further down:

```
10 MODE 1
20 VDU28,0,25,15,6
30 PRINTTAB(0,0)"Tab 0,0 starts here"
40 PRINTTAB(3,3)"Tab 3,3 starts here"
50 PRINTTAB(11,11)"Tab 11,11 starts here"
60 VDU26
70 END
```

In the program for a woman's age, text windows served two different functions:
the first meant that the instructions on the first screen were visible all the time input was wanted; it also was made small enough that only one question was on the screen at at time - less to take in, and

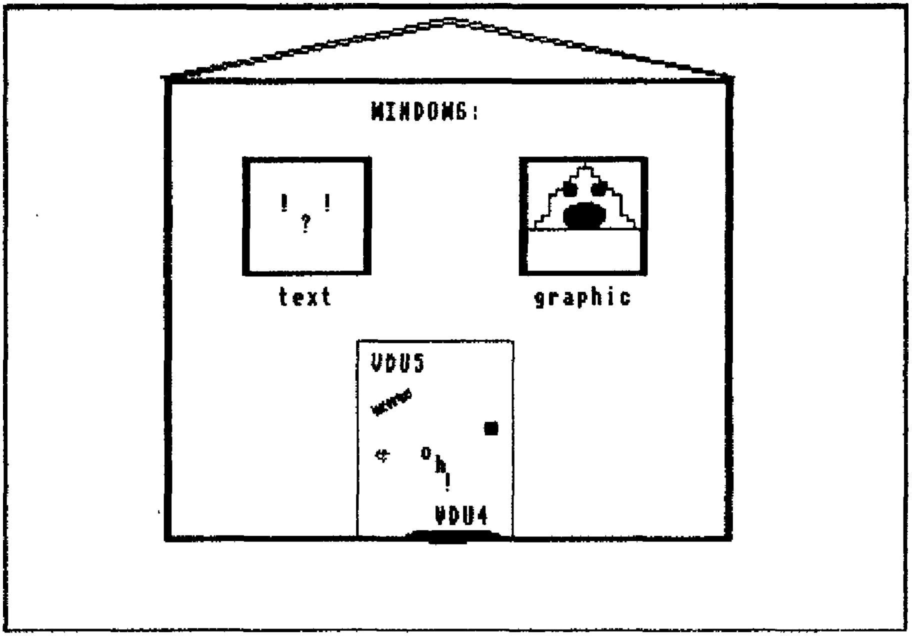
harder for others in the family to sneak a look and calculate Aunt Maud's real age. The second window for PROCbirth was a way of changing the background colour for that section of text only.

THE MAIN LISTING:
20 define character 223 - modern computers have mice as well as windows, so we'd better keep up with them.
40 prints a mouse at what will be bottom left of a new text window
50 prints a mouse at what will be the top right of the new text window
70 define the window properly
80-140 do 10 times:
90 Choose a colour
100 draw a triangle anywhere on the screen (shades of the Welcome booklet)
110 give you a second to take it in
120 Print the loop number
130 Print some text.
In these 2 lines notice that as the text scrolls upwards, so do any parts of triangle that had intruded on the text window (the graphics window is the whole screen)
150 Clears the text screen
160-180 prints 20 mice with a space after each one.
170 could also have been written as: 170 PRINTCHR$223;" ";
Notice how well-trained these mice are, staying within the boundaries!
210 set up a different text window
220 Clears the latest text window - notice it clears any graphics in its area, but doesn't affect the mice - as far as the computer is concerned, those mice are in a graphics area. Their text window no longer exists, so it can't be cleared by CLS. Try CLG instead.
230-240 print up some text to show the constraints of the new text window.
250 Undefine the text window! Doesn't appear to do anything, but leave it out then try to LIST your program after RUNning it!!

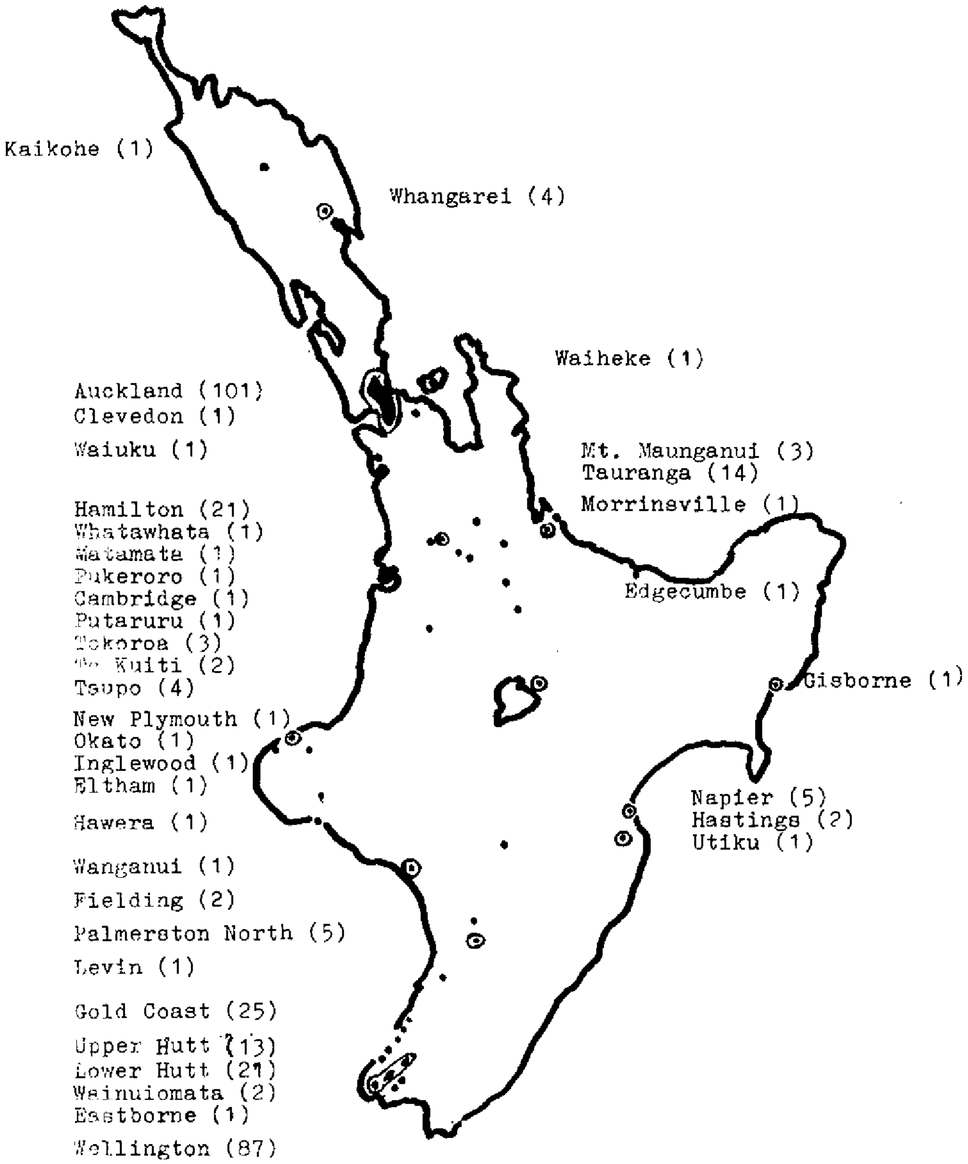
EXPERIMENTS:
Change 90 to GCOL0,RND(3)
150 CLG: what effect?
Define a graphics window before line 80

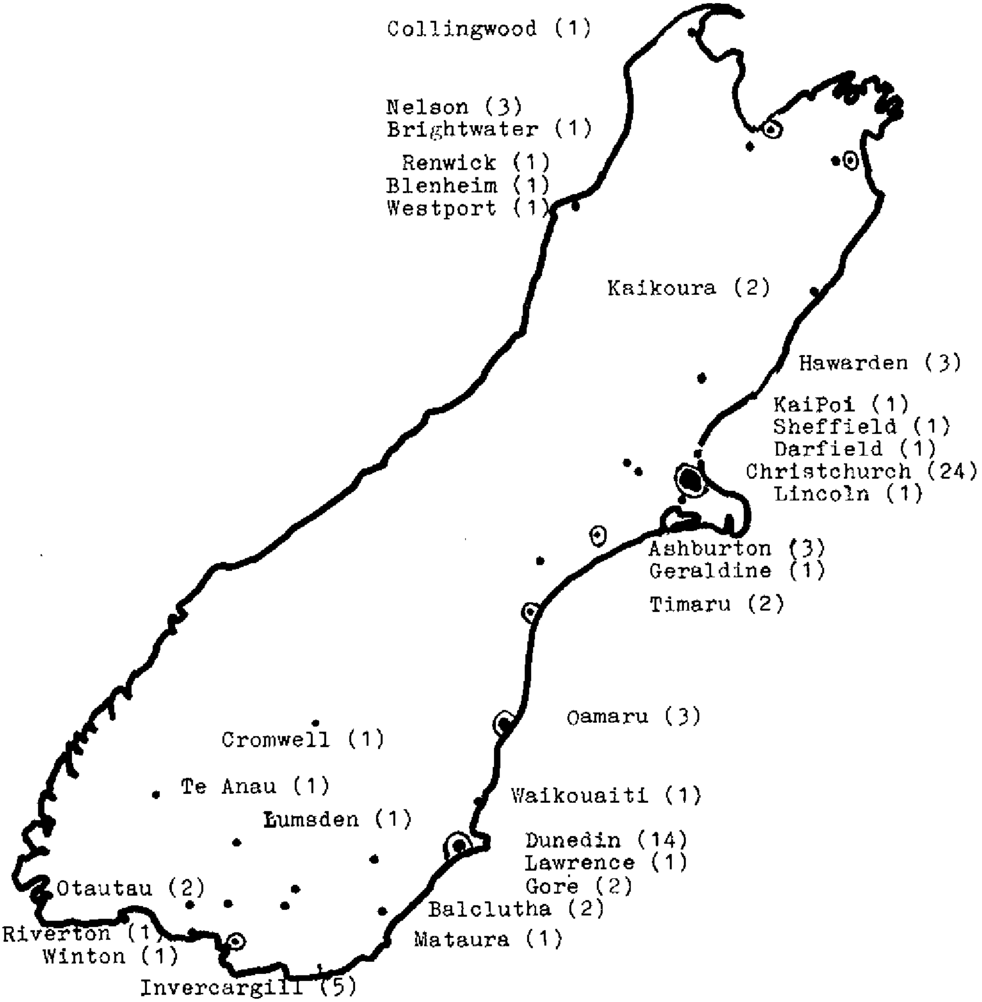
```
10 REM text windows
20 VDU23,223,&18,&BC,&5E,&FF,&01,&0E,&10,&00
30 MODE 1
40 PRINTTAB(15,14)CHR$223
50 PRINTTAB(25,11)CHR$223
60 wait=INKEY(100)
```

```
 70 VDU28,15,14,25,11
 80 FOR I%=10 TO 19
 90   GCOL3,RND(3)
100   PLOT85,RND(1279),RND(1023)
110   wait=INKEY(100)
120   PRINT;I%
130   PRINT"mice are"'"coming soon";
140   NEXT I%
150 CLS
160 FOR I%=1 TO 20
170   VDU223,32
180   NEXT I%
190 VDU7
200 wait=INKEY(200)
210 VDU28,0,25,15,6
220 CLS
230 PRINT"Got another text"''"window 'ere,"
240 PRINT''"Fred,  sickening"''"aint it?"
250 VDU26
260 END
```



| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | Y | N | T | A | X | | | U | | T | O | O | B | I | G | | S | T | O | P | | | | O |
| C | | | A | | | E | V | N | T | V | | N | | | E | | O | | | I | N | S | T | R |
| R | I | | B | F | | C | | C | | | | | P | I | T | C | H | | C | | | I | | |
| O | | B | | | G | O | T | O | | A | E | | | N | | H | | | O | P | E | N | I | N |
| L | D | A | | R | | N | | M | O | T | O | R | | T | | R | S | | U | | | | | E |
| L | | U | | O | | E | | M | | | R | | I | | L | | | | N | | I | F | | X |
| M | O | D | E | M | | T | | I | R | Q | | I | N | D | I | R | E | C | T | I | O | N | | T |
| O | | | | | D | | V | T | | | 1 | | T | | S | | | L | | | | | | |
| D | | S | A | V | E | | | T | E | L | E | T | E | X | T | | | S | O | U | N | D | | G |
| E | X | P | | | F | F | E | E | | O | | | R | | | | L | | | | | Y | | C |
| | | O | | | | S | | D | | G | | A | R | G | U | M | E | N | T | S | | N | | O |
| D | | O | P | T | | | U | L | A | | V | D | U | | G | | T | | R | | C | A | L | L |
| E | R | L | | A | | | | O | | | | C | P | U | | | | N | U | L | | M | | |
| G | | | A | N | A | L | O | G | U | E | | | T | | J | I | M | | E | | V | I | A | |
| | T | | | | | D | | I | | R | | E | | U | | N | | | | | A | C | S | |
| F | I | E | L | D | S | Y | N | C | H | R | O | N | I | S | A | T | I | O | N | | L | | C | R |
| F | | N | | | P | | A | A | | | | D | | E | | E | | S | | O | | S | | E |
| D | I | V | | | C | | | R | | P | | | | R | | R | E | G | I | S | T | E | R | S |
| A | | E | | 8 | | P | A | R | A | L | L | E | L | | | F | | B | | | | R | | E |
| | C | L | E | A | R | | | A | | | O | | | P | H | A | | P | | N | M | I | | T |
| | | O | | | | | | Y | | C | M | P | | | E | C | | B | | | | A | | |
| C | A | P | S | L | O | C | K | | | | E | | P | I | X | E | L | | | S | I | L | L | Y |
| L | | E | E | | | L | | | D | I | M | | T | | | | | E | X | T | | | N | |
| G | | | I | | E | V | A | L | | | | | R | E | A | D | | 6 | | R | O | L | | |


Kaikohe (1)
Whangarei (4)
Waiheke (1)
Auckland (101)
Clevedon (1)
Waiuku (1)
Mt. Maunganui (3)
Tauranga (14)
Morrinsville (1)
Hamilton (21)
Whatawhata (1)
Matamate (1)
Pukeroro (1)
Cambridge (1)
Putaruru (1)
Tokoroa (3)
Te Kuiti (2)
Taupo (4)
Edgecumbe (1)
Gisborne (1)
New Plymouth (1)
Okato (1)
Inglewood (1)
Eltham (1)
Hawera (1)
Napier (5)
Hastings (2)
Utiku (1)
Wanganui (1)
Fielding (2)
Palmerston North (5)
Levin (1)
Gold Coast (25)
Upper Hutt (13)
Lower Hutt (21)
Wainuiomata (2)
Eastborne (1)
Wellington (87)


Collingwood (1)
Nelson (3)
Brightwater (1)
Renwick (1)
Blenheim (1)
Westport (1)
Kaikoura (2)
Hawarden (3)
KaiPoi (1)
Sheffield (1)
Darfield (1)
Christchurch (24)
Lincoln (1)
Ashburton (3)
Geraldine (1)
Timaru (2)
Oamaru (3)
Cromwell (1)
Te Anau (1)
Waikouaiti (1)
Lumsden (1)
Dunedin (14)
Lawrence (1)
Gore (2)
Otautau (2)
Balclutha (2)
Riverton (1)
Mataura (1)
Winton (1)
Invercargill (5)

## HELP WANTED

Allister J. writes : We have just upgraded to discs and find it really great but for the smaller memory. With some of my larger BASIC programs I get the dreaded "BAD MODE" and "NO ROOM" errors. Sometimes both at once. Yet there is almost sufficient room to have the program in there twice in MODE 1! Even when i issue a *TAPE command and shift it down to &E00 I still get it jamming up. Also sometimes after a '*' command and a soft Break, any disc accessing commands simply set the disc spinning and jam the computer, leaving the disc going.

Does anyone have a fill routine for use in MODE 7 ? The excellent book "Graphics on the BBC Microcomputer" by Neil, Pat and Andrew Cryer contains a fill routine which appears to have been converted to BASIC from machine code. The routine is good for simple shapes, however due to its recursive nature it gobbles up memory like it was going out of fashion. It can be improved by chaging the first LOCAL statement to omit the list of variables. Try entering it as listed in the book, then, in MODE 1, LIST the program and type PROCfill(0,0) and before pressing RETURN, hold the CTRL key down and press R, @, A, M. You wont need to press RETURN as the CTRL M does the same thing. Now watch how far it gets before it issues a "No Room" error. Next change line 240 as I said before and repeat the sequence starting with MODE 1. You will see that it goes on and finishes the job this time. However, the second "LOCAL" statement is still goblling up bytes and cannot be changed as the routine will not work properly. With the improved version I have still found occasions when it will run out of memory, especially so with a MODE 2 disc based program!

How do you hook up a chess board to the Beeb? Through the User port? After making up the activity board in August 'Acorn User', it occurs to me that it may be possible to set up a chess board in a similar way. The activity board used dry reed switches; when a magnet was passed over one, it shut, and the computer knew "where" the magnet person was. Can someone explain how to set up a 64-location set of switches; and is there something cheaper or better than the switches mentioned above. At this stage, I am mainly interested in playing through a chess game and getting an accurate printout of the moves, though obviously the same setup could be used for other board games. Also I would like the answer in simple English - I don't know enough to read circuit diagrams, but I do know how to use a soldering iron.

### ECONET AT HOME?

With a second processor in the house which has the DFNS chip that Econet also uses, is it possible to set up programs at home and get them to run without further change on a Level II Econet at school? Or should a disc be prepared at school, then be taken home for work which will then work back at school again without running a conversion program.

# HINTS...

OS, BASIC, Beebs, Elks, what-have-you

How do you know what you have got?

BASIC? OS?

A lot of people seem confused as to what they are, and what they do - I've even had someone 'regret' to say that they had only an OS1.2 and not version 2!

They are actually separate ROMs. Most, if not all Beebs sold recently have BASIC II ROM (Basic language version 2), and OS 1.2 ROM (*referred to often as a 'series 1* Operating System').

All Beebs and Elks must have an operating system to function in any way, but you can remove the BASIC ROM and replace it with a suitable alternative (eg VIEW), and still run the machine. Of course, it won't be able to speak 'Basic' *any more*, but it *will* be a *wonderful word* processor.

Electrons have Operating System (OS) 1.2. Beebs may have OS 0.1, 1.0, 1.2.
*HELP <RETURN> will tell you which version you have. If you have a Disc filing system (DFS), or other ROMs, these will also be displayed on the screen. Some ROMs, such as BASIC will not show up! *HELP with a co-processor active will show which version of Tube software they have.

WHICH BASIC?

Aha! The copyright date is the clue. Switch on your machine, or do <CTRL> BREAK.
type: REPORT <RETURN>
If the year 1981 appears, you have BASIC I. If it's 1982, you have BASIC II.

ELK vs BEEB BASIC differences:

The Elk's having the equivalent of the Beeb's Basic II and OS 1.2 has caused a bit of confusion when people make comparisons between the two machines, not realising that their Beeb may have Basic I. There are, however, genuine differences, such as the Elk's lack of Mode 7, and the reduced SOUND and ENVelope implementation. Also, *FX calls to the operating system are dependant on the hardware features of the Elk system.

## LOST MEMORY?

RAM is always inadequate at the best of times. Disc users will often be painfully aware of another 3K missing: grabbed by the DFS - Disc Filing System. There are ways of grabbing back some or all of that 3K though.

1) Resetting Page. On tape machines, PAGE is &E00. On disc machines it is further up the "book" of memory at PAGE &1900. Most of the distance between these two numbers is only used if you are accessing other files from your program. Thus for a simple program that is not a data-base, spreadsheet, wine list,... this space is wasted. It can be claimed by typing in " PAGE=&1100 ". This still isn't back down to E00, but means you will be able to load and save programs to disc without trouble. If your program only needs one disc file open at a time then PAGE can be reset to &1500. For some programs this method gives enough memory for the program. If not...

2) Downloading:
This is a way to load your program from disc, then move it down to Page &E00. Of course, it then means that you cannot use the disc system without corrupting the program - a trial when you have typed in a program but haven't found all the errors, as it means you have to reload the program from disc to find the error in the listing, as well as saving it before running it again. However, it does give you as much memory as if you were still using tape! And you can still use TAPE files!

There have been various routines published in UK magazines. For those who have missed them (or are too broke), here is one from 'Micro User' Nov '84 p.192. It is intended to be typed in at the start of your program - lines 1 to 5. It works by checking PAGE. If it isn't at rock bottom already (ie &E00), it defines a function key to turn your machine into a tape-based one, reset PAGE and shift the program down from &1900 (or wherever) to &E00. It also deletes these lines of code. After defining the function key, it RUNs, and presses the function key for you!! If you don't like function key 0 being redefined, but perhaps key5, change line 3 to read *FX138,0,133

```
1 IF PAGE=&E00 THEN 5
2 *KEY0*TAPE¦MD%=PAGE-&E00:FOR L%=PAGE TO TOP STEP4:
!(L%-D%)=!L%:NEXT:PAGE=&E00¦MOLD¦MDEL.1,4¦MRUN¦M
3 *FX138,0,128
4 END
5 REM  the rest of the program
```

# EDITING BASIC IN VIEW

David Fung

The steps required to edit basic programmes in VIEW are as follow:
With the listing in the computer

| | |
|---|---|
| 1. SAVE "PROG" | Unedited "PROG" saved |
| 2. *SPOOL S_PROG | |
| 3. L. | |
| 4. *SPOOL | S_PROG saved |
| 5. *WORD | |
| 6. LOAD S_PROG | That is how you get it into VIEW |
| 7. ESCAPE | LOAD not READ |
| 8. DO YOUR EDITING | |
| 9. ESCAPE | |
| 10. SAVE V_PROG | V_PROG saved in VIEW format |
| 11. *B. | |
| 12. *EXEC V_PROG | This types it out as from keyboard |
| 13. L. | L. tidy things up |
| 14. SAVE "E_PROG" | Edited version "E_PROG" saved |
| 15. CUP OF TEA | You've earned it |

## ONE-SIDED DISC DRIVE?

There are at least two red faces around - people who bought a disc drive and assumed they had only one side to it. Or didn't realise there were two sides on their drive. How do you find out? Ask the dealer who sold it to you! Or, the less foolish-looking method is:
In the privacy of your home, take the unformatted disc and try to format drive 2 as well as drive 0.
or
put in someone else's disc which has programs on both sides, and try to get a catalogue of both sides.
*. <RETURN> will tell you what's on the top side, and
*.2 <RETURN> will display contents of the bottom side or give a 'rude' message such as 'disc fault' if you have only a single-sided drive.

REM *. is an acceptable abbreviation for *CAT 0

# SORT REPORT - PART ONE.

Steven R.

Most people have occasion to sort things sometimes, like Christmas card lists for instance. But how do you do it on a computer? This is the first of a series of articles about the problem. In each article, I will describe a method of sorting, with it's advantages and disadvantages, and include a listing (that probably works). There will either be a running comparison in each article, or a summary at the end of the series, depending on whch seems to work best next time.

Anybody who already knows about a sort described in any of these articles is welcome to comment on (correct?) them. I am not the world's best at describing things and I'll probably leave somethng out or manage to confuse you somewhere.

Note: If you are confused, so is someone else, so send Bruce a letter about it, or phone me, so that I can correct it next time (Anton knows my number. It's also in the listing in the August Beeblet last year.)

## BUBBLE SORT

This is the simplest sort, and to my knowledge the first designed for a computer.

It simply runs up through the array of numbers, swapping any that are out of sequence. It does this several times until everything is in order.

A typical program looks like this:

```
1000 DEF PROCbubble_sort
1010 FOR stop% = MAX TO 2 STEP -1
1020   FOR sub% = 1 TO stop%
1030     IF element%(sub%) > element%(sub%+1) THEN PROC
1040   NEXT
1050 NEXT
1060 ENDPROC
```

I have left PROCswap for you to do.

Taking lines 1020 to 1040 first: each element is compared to the next element, and if it is greater that it, I.E. out of order, the two are swapped. This is done until we reach the stop% element.

Lines 1010 and 1050 form an outer loop, that decreases stop% by one each time until it reaches 2. This is done because at the end of every run up through the array, the last two elements must be in order. Because of this, we don't need to check them next time. When stop% reaches 2, the whole array must be in order. Try it!

This works, but there is something that can be done to improve it. What happens if the array is already sorted? The program goes through the array time after time, checking for elements that are out of order, even though there aren't any!

To make it stop when the array is in order, even if stop% hasn't got to 2 yet, we must add another two lines, and change one, like this:

```
        1015 noswap% = TRUE
        1030       IF element%(sub%) > element%(sub%+1)
THEN PROCswap:noswap% = FALSE
        1045 IF noswap% THEN stop% = 2
```

The program now looks like this:

```
        1000 DEF PROCbubble_sort
        1010 FOR stop% = MAX TO 2 STEP -1
        1015   noswap% = TRUE
        1020   FOR sub% = 1 TO stop%
        1030       IF element%(sub%) > element%(sub%+1)
THEN PROCswap:noswap% = FALSE
        1040     NEXT
        1045   IF noswap% THEN stop% = 2
        1050   NEXT
        1060 ENDPROC
```

This checks to see if a swap is made in any particular pass through the array. If not, then the array must be sorted, so stop% is set to it's finishing value so that the outside loop will stop.

In this example, element% was used as the array, but real numbers and strings can be sorted just by changing the type of the array. Nothing else needs to be changed, because the beeb can compare strings as easily as numbers.

Advantages:

1. This sort sorts arrays that are already in order very quickly.
2. No extra room is needed besides the array.

Disadvantages:

1. The bubble sort takes a long time to sort large arrays because of doing so many comparisons and swaps.

Next time, another sort, and a comparison of it and the Bubble sort's virtues.

*********************************

A note about "Files: Part Three", with apologies from Steven R.

Definition of POINTER: A number that gives the location of something, somewhere else. Frequently either the number of the byte in a file where something starts, or the address in memory where something can be found.

**APPLIED ENGINEERING AND TECHNOLOGY LTD**

**3 HALLEYS LANE** **phone (04) 850-800**
**WELLINGTON**

**aptec**

**FOR BBC AND ALL TYPES OF MICROCOMPUTER SERVICING**

# MERLIN SCRIBE

$279.95

( Supplied for review by Whitcoulls )

Reviewed by Neil G.

The two dominant word-processing programs for the BBC are WORDWISE and VIEW. In the former, text is entered into the computer using the 40 column MODE 7, with formatting commands sprinkled throughout the document. These commands specify such things as page length, left margin settings, whether justification should be done, or that text should be centered on the page. The final form of the document can only be seen by "previewing" it to see the effect of the formatting commands, or by printing it out. By contrast, VIEW is in the "what you see is what you get" mould. Text can be entered in either 40 or 80 column MODEs, but the formatting takes place on the screen itself, allowing almost the final form of the document to be seen.

However, since a WORDWISE document is always contained in memory, there is a limit to its size - around 28 thousand characters. Furthermore, previewing of the final form in 80 column mode is impossible if more than a few thousand of these are used up. VIEW is also usually used to edit documents contained wholely within memory, though with disk systems it can be used in a continuous processing mode where text is read in sections from one disk file, edited, then written out to a new disk file and the next section read in, and so on. In this mode, however, you cannot go backwards, so it is not terribly convenient.

MERLIN SCRIBE also falls into the "what you see is what you get" category. However, it is probably the first BBC word processor which makes use of the random access properties of disks. When a document is initially created, both the page dimensions (e.g., 55 lines of 65 characters for about an A4 page) and the number of pages must be specified. SCRIBE then creates a disk file of sufficient size to hold the entire document, including blanks to fill out each line. The document can then be edited a page at a time, with that page being held in memory and the rest on disk. If you move on to another page, the current one is written back to disk (even if it hadn't been modified), and the new one read in to be edited.

Having to specify exactly the size of the document in terms of page size and number of pages is a bit of a nuisance. Fortunately, if you change your mind you can use a utility provided to create a new document with different sizings and copy the text to that. On the plus side, you can have a document limited only by available space on the disk.

SCRIBE comes with an 8K EPROM to plug into a ROM socket in your BBC (if you have any left), a disk containing additional utility programs, an overlay for the function keys, and a 30 page manual. The utilities on the disk are indispensable, as they perform vital functions such as creating a new document file, printing out a SCRIBE document, and providing the full SCRIBE menu. The disk also contains a !BOOT file which should be used to start up SCRIBE in preference to the *SCRIBE command, which doesn't really do all the initializations that should be done (why not?). So, to start up SCRIBE, you insert a copy of the utilities disk (keeping the master somewhere safe), press SHIFT and BREAK, and are presented with the main menu containing 6 options - you can edit, create, print or change the document, or select utilities or disk commands.

Let's suppose you have already created a document by specifying its size, and now wish to edit it. You select EDIT from the main menu, then give the drive number and document name. Page 1 will come up on the screen (in 80 column mode, though you can use 40 columns if you really want to), with a status line at the top of the screen. This gives the document name, current mode (i.e., whether INSERT is on or whether characters will overtype existing ones, and whether JUSTIFY is on so that text will automatically be right-justified), and the current cursor line number, position of the line, and page number. The cursor is moved around using the arrow keys, but I found that it moved much more slowly than for WORDWISE and was initially disappointed not to find a quick way of moving in bigger jumps such as by words, or all the way to the start or end of a line. (For example, it takes at least 6 seconds for the cursor to move from left to right across a 65 character line with the right arrow key held down.) Eventually I discovered that the function key f8 can be used to move the cursor to the left hand side of the line, or up or down half a page by pressing it with CTRL or SHIFT - unfortunately this information is missing from the function key overlay.

Characters typed at the cursor position will normally be inserted, with existing text moved a line down until you have finished the insertion when it will be moved back into place. This is a little disconcerting compared to WORDWISE, and makes it difficult to match up the insertion with existing text. Words entered are automatically "word-wrapped", being moved to the next line if they won't fit on the current one. There doesn't seem to be any means of deleting the character (or word) at the cursor position - you need to move one character beyond it and then press DELETE to remove the character to the left of the cursor. This can get tedious.

To delete larger sections of text, you can put the cursor at the start of the section, press f1, then use the arrow keys to move the cursor to the end of the block to be deleted (it will be displayed in reverse video). *Amazingly, if you accidentally move the cursor too far,* you can't just back up, but need to abort the operation (by pressing ESCAPE) and start again. What seems particularly poorly planned about this is that if you move the cursor a line at a time *by* hitting the *down* arrow, you select the first character of the next line for deletion! This makes it *easy* to go too far, and have to start again. *Anyway*, once you have selected the text to delete you just press RETURN and it goes away.

Moving a block of text, or making a copy of it, is handled similarly. In fact, the same function key (f1) is used, but with SHIFT pressed for move or CTRL for copy. I like the consistency. What I don't like is the limited size of the block of text you can move or copy - *no more than 9 lines!* This is because the text is buffered in memory, and there is not a lot left when using a MODE 0 display and storing the current page of text. I am surprised that use was not made of the disk to store the text *to be moved - 9 lines is* a severe limitation.

A nice feature is that text can be underlined, and this will show up on the screen. Again, you mark the start of the section to be underlined, move along to the end, and then press RETURN to mark the end. Again, if you accidentally go too far with the cursor, you have to abort the operation and start again. If you insert extra text into the underlined section it will not be underlined. If you copy a section of text which includes an underlined section it will not usually justify properly because it is somehow "glued" (their term) together.

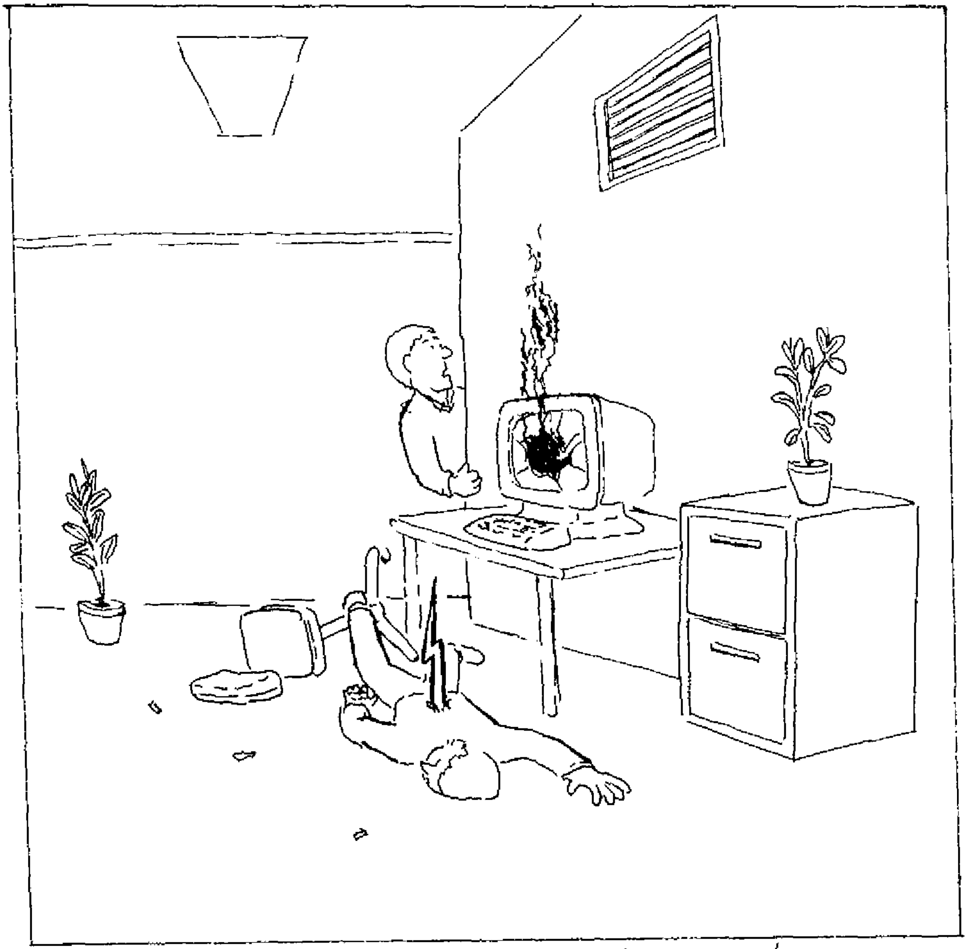
Other features include the usual commands to set left and right margins, specify that text should be centred, and so on. The formatted text is shown on screen as it will be printed out. Control characters for a printer can be inserted directly into the text, and will be displayed as little two digit numbers taking up just one character position, with justification automatically compensated for.

You can search (forward only) for a text string of up to 30 characters, and optionally replace each occurrence with a different string. As with WORDWISE and VIEW, the search is unavoidably sensitive to case, so if you search for Scribe you will not stop at a SCRIBE.

Printout control is extremely comprehensive - on selecting print from the main menu you get a new menu of 18 items which can be amended at will, and the values stored together with the document so that having got it right you needn't bother again. The items covered include whether the printer is parallel or serial (including baud rate), line spacing, whether a line feed should be output as well as a carriage return at the end of each line, margin settings, print headers and footers, and so on. A unique feature is the ability to redefine up to 5 characters. This allows you, for example, to redefine [ to give the control characters for start of bold print (plus a space to ensure that justification is retained), ] for the end of bold print, and then use [ and ] within the document to enclose text that should be printed bold.

The utilities include access to operating system (*) calls, counting words in a document, extracting pages from a document, or EXECing a document. The latter allows editing of a BASIC program source via a rather complicated procedure, and also allows the SCRIBE document to be put into a file that WORDWISE can understand (hence this review - which was entered using SCRIBE and then converted to WORDWISE).

In summary, MERLIN SCRIBE is a comprehensive word processor which makes better (though not the best) use of disks than other BBC word processor programs, but has some rather strange limitations which could be annoying until you get used to them. Although comprehensive, SCRIBE does not have some of the "professional" features of VIEW such as macros, microspaced printout, and the ability to print opposing pages. The bottom line is that SCRIBE, kindly provided by Whitcoulls for this review, costs $279.95. It could be worth it if you were interested in processing large documents in one chunk and were willing to put up with some of the idiosyncracies.



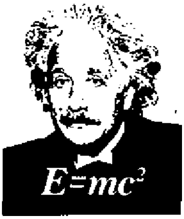
George! ...that voltage spike problem is back again.

EINSTEIN SCIENTIFIC LIMITED
177 Willis Street
WELLINGTON.

CHECK EINSTEIN'S EXTENSIVE RANGE OF PRINTERS AND SOFTWARE FOR THE BBC & ELECTRON MICROS.

**WELLINGTON**
177 Willis Street. Phone Raju, Michael or Malcolm 851055 or 844353.
**PALMERSTON NORTH**
154 Broadway Avenue. Phone Richard 64108.
**NEW PLYMOUTH**
Corner King & Egmont Streets. Phone Bill or Sally 85528.
**CHRISTCHURCH**
Shop 41, 1st floor, Cashfields Mall Phone Mike 66442.
**HAMILTON**
inside DIC, Garden Place. Phone Maxine 81969.

Einstein
SCIENTIFIC LTD



The computer world made simple.

# AVIATOR By Acornsoft

Reviewed By A. Beeston

## CONTENT.

The Aviator package contains a tape, a booklet, a map and a sheet showing the keyboard functions.

The version I have of Aviator is on tape, however the book shows that a disk version must also exist.

The booklet outlines Aviator in great detail. It gives an outline of the display and controls of the simulator. Information each aspect of flying from taking off to aerobatics is given, together with details on scoring points by destroying aliens.

The map of the terrain shows what terrain the computer recognises. It includes many trees , hills, the runway, a river with a bridge, a lake and a city.

The last part of the package is the sheet showing the keyboard layout of the simulators controls. This is an invaluable aid when first learning to use the simulator as even if you have a joystick the keyboard must be used in some cases.

## OBJECTIVES.

The pilot is set the task of flying a Spitfire aeroplane. The objective of aviator is to gain points by shooting aliens, landing the plane and stunt flying. Stunt flying consists of flying through the city below the level of the lowest building or flying under a bridge.

The pilot decides before taking off if he wants simply to fly the plane or if he also wants aliens to attack. If aliens are present then the pilot must shoot them down before they descend on the city. The game ends when you crash or when aliens destroy the city.

## OPERATION.

Input to control the simulator is by keyboard and joystick. All functions can be used by pressing keys, while some functions can also be controlled with a joystick. The joystick provide the trigger for the plane's gun and operates the elevators and ailerion controls.

## FLIGHT REPRESENTATION.

Aviator displays all images in black and white line drawings, and text. The screen is divided into two parts, representing cockpit instruments and views through the front of the cockpit window.

The computer graphically represents the cockpit instruments. As a result the accuracy of readings taken from these instruments is very limited. This makes it necessary to fly by interpreting what is seen out of the cockpit as well as using the instruments.

The instruments are standard needle and dial instruments as would be found in a real Spitfire type aircraft. However there is also a radar screen which would not normally be found in a Spitfire. The radar shows the plane in the centre of the screen and shows the runway's position in relation to the aircraft. Aliens also appear on the radar. The position shown by the radar are very approximate and final navigation is done by interpretation of what is seen through the cockpit window.

The view through the cockpit attempts quite successfully to represent three dimensional objects at changing distances. However the view is what a pilot would see looking steadily forward out of the cockpit. As a result the screen often shows no terrain and to see the ground the plane's attitude must be changed rather than the pilot's.

## GAME RESPONSIVENESS

The game is very responsive to keyboard and joystick controls. In particular the joystick takes some getting use to because at high speeds very small changes in the position of the joystick have a great effect on the attitude of the plane.

The only unresponsive control is the gun as only one shot can be in the air at any one time. It takes a while for the shot' to reach its maximum range. As a result if a shot misses there is a considerable delay before another shot can be fired.

When an alien is hit it creats a large amount of local turbulence. If the plane is close to the alien when the latter explodes it will go out of control and a lot of effort is required to prevent a crash. It is possible to shoot the aliens at a distance, but with fixed position guns great control of the aeroplane is required to hit a small target , even with the assistance of the gun sight provided by the simulator.

## IMPRESSION

I like this simulation : it is very challenging, the most realistic and hardest I have seen for the BBC. I have tried Jumbo by from Molymerx, 747 from Doctor Soft, Spitfire from Alligator, and F for Freddie by Kansas City Systems and feel that Aviator by Acorn is the best flight simulator I have seen for the BBC.

## THE POWER OF VDU5 FOR BEGINNERS TO GRAPHICS

Jeff H.

Note the paragraph in the user guide page 75 below the example program which says that after a VDU5

(1) the screen will not scroll -(a stable picture)
(2) characters will superimpose -(allows more than one colour in a character)
(3) colour is selected by GCOL -(a more powerful command than COLOUR because the extra parameter allows mathematical control of collisions)

Other properties of VDU5 are:

(1) Only the graphics window is used.
(2) Printing is at the graphics cursor which ignores the TAB(x,y) command but responds to the more precise MOVE command. (After each character the graphics cursor moves one character space without a newline. Print statements can be made with VDU followed by the appropriate no. for user-defined or pre-defined character.)

It is useful to emphasise the paragraph on page 75 because twice in the U.G. (p.74 & 173) the VDU5 command is introduced as a device for precise positioning of text whereas in fact it is THE command which umleashes the graghics power of a machine famous for its graphics power.

Apart from this uncharacteristic modesty the user guide does a good job of text(p.67); line and point graphics(p.160); colour(p.162) and character creation which are all needed by the budding TV artist.

## CLASSIFIED ADS
**************

### DEALS AND DISCOUNTS

Send stamped addressed envelope for details of the deals negotiated by the Wellington Branch for UG members. Please mark your request for the attention of Jill Armstrong, C/O the UG Box 9592 Wgtn.

FOR SALE:

### MICRO COVERS

Designed to fit the BBC micro. Attractive opaque reinforced plastic, $5-50 posted. To Iain, Box 26078, Epsom, Auckland 3.

### CASSETTE TAPES

C10s, boxed, leaderless - available from the UG in multiples of 20 : $26-00 plus $2-00 p&p.

### FLOPPY DISCS

3M Brand 5 1/4" DSDD discs - available from the UG ex-stock in (cardboard) boxes of 10: $55-00, which includes p&p.

### RGB

Converted 18" TV. Better resolution than many commercially available RGBs. $400-00 ono.
Phone Allister 399-696 (after 6 pm Weekdays).

### ACORN ELECTRON

As new (6 months but rececently overhauled by Barsons) + some software and extra book. Genuine reason for sale - have upgraded to BBC B. $700-00 ono.
Phone John Sutherland wk 726-499 x8759 or home 886-768 after 6pm.

### DISK DRIVE

One 40-track dual 2x 100K disk drive, cased, cables, manual, and utilities disk. Price $550-00 ono. Write to Barry Borman, PO Box 9293 Wellington or phone WN 898-855.

### ELITE - DISK VERSION.

Jeremy Brew wants to sell his disc version of Elite (his upgrading plans have been upset). He would like the cassette version (part exchange?). Write to Box 54-137, Plimmerton or ring 399-253.

# HALL OF FAME

| PUBLISHER: | GAME | HISCORE | SCORER |
|---|---|---|---|
| Acorn | Arcadians | 15,000 | Felix G. |
| | Aviator | 2,500 | Steven J. |
| | Countdown | 227 | Tim R. |
| | Elite | Above Average | Ceri L. (age 9) |
| | Elite | Elite | Robert N. |
| | Hopper | 27,196 | Jenny B. |
| | Meteors | 24,090 | Duncan's flatmate |
| | Monsters | 93,560 | Linsay F. |
| | Planetoid | 413,875 | Peter J. |
| | Rocket Raid | 140,777 | Mark R. |
| | Snapper | 250,000 | Susan |
| | Starship Command | 9,495 | Robert N. |
| Alligat. | Bug Blaster | 37,642 | Steve W. |
| Aardvark | Zalaga | 56,060 | Geof P. |
| | Frak | 46,900 | Geof P. |
| | Monsters | 175,110 | Geof P. |
| Bugbyte | Twin Kingdom | 385 | A. Beeston |
| Dr Soft | '747' | 205 | Steve W. |
| | Gorf | 18,648 | Steve W. |
| FG Soft. | Mutant Defender | 6,900 | Steve W. |
| Level 9 | Colossal Cave | 687 | Steve W. |
| | Dungeon Adv. | 330 | Anne et al |
| | Lords of Time | 925/1000 | Steve W/Kerry K. |
| | Snowball | 900/1000 | Steve W./Dianne |
| Prog.Pow | DangerUXB(Lvl 1) | 286,690 | Steve W. |
| | Felix the Factory | 7,000 | Dianne S. |
| | Frenzy | 915,468 | Dianne T. |
| | Killer Gorilla | 91,000 | Susan |
| | Super Invaders | 19,610 | Brett A. |
| Sup.Soft | Alien Dropout | 24,200 | David E. |
| | Q*BERT | 30,260 | Dianne S. |
| | Roadrunner | 70,600 | Lois S. |

HALL OF FAME

# USER GROUP MEETINGS

AUCKLAND: -meets 2nd Wednesday of the month, 7.30pm, Conference Rooms, Auckland Secondary Teachers College, 60 Epsom Ave, Epsom. ph Dave 770-630 x 518(bus) or Kerry 695-355.
CHRISTCHURCH: -fortnightly workshops, Monday 6.30pm at Hagley High School. ph. Michael 582-267.
DUNEDIN: -Phone Martin McDowall 42-031
HAMILTON: -last Thursday of the month 4.45pm Waikato Tech B-block staff-room. Ph. Peter, Hamilton 392-500 x897(bus) or Alison, Morrinsville 6695.
HAWKES BAY: -Alternative months Hastings & Napier. Contact Kendall Napier 435-624, Bob Taradale 446-955, Mitch Hastings 778-235.
INVERCARGILL: -Phone Evan 330-444 Invercargill.
PALMERSTON NORTH: -Phone Tom Skinner 83-009(wk)
TAUPO: -Contact Dave 84-215.
TAURANGA: -every second Wednesday. ph. Chris & Jane 65-076.
TIMARU: -Phone Lloyd van der Krogt Timaru 80-082.
TOKOROA: -meets first Friday of each month, Tokoroa High School, 8pm. Phone Brian 67-625 Tokoroa.
WELLINGTON: -meets last WEDNESDAY of the month, 7.30pm, Royal Society Lecture Theatre, Turnbull St, Thorndon.ph. Anton 286-289 or Warren 787-005.
OTHER CENTRES: - let us know and we will publish details here.

## THE NEWSLETTER:

CONTRIBUTIONS:-most welcome. Listings should be sent in on tape or disc. (Tape - 2 copies, one at 300 baud please). A 35c stamp gets your tape / disc returned too. Please include written explanation of listing; text only material welcome too.

DEADLINES:- For a particular issue, the last day for material is the last day of the month prior to publication.

ADVERTISING:- Rates are $20 per half page (camera ready), deadline as above.

Copyright (c) **THE BBC/ACORN COMPUTER USER GROUP OF NZ INC.**

Published monthly except January and mailed to financial members.

## MEMBERSHIP:

Membership of the users group is on payment of an annual subscription from April to March. For the 1984/5 year it is: $30.00, plus a $5.00 joining fee for new members, plus $5.00 for Auckland & Wellington members as a local branch levy, which entitles attendance at meetings as detailed above. Renewing members get all back issues of the current year; New members also get back issues to v.1 #1 Jan '84.

THE ACORN/BBC COMPUTER USER GROUP OF NZ INC, P.O.Box 9592, Wellington.

Reproduced by ROBERTS PRINTING Co. Ltd, Wellington.